Advanced R

EANBiT RT 8th July 2022

David Kiragu Mwaura

Tidyverse

- Powerful collection of R packages that
- Contain data tools for transforming and visualizing data
- Core package include: dplyr, ggplot2, tidyr, readr, purrr, tibble, stringr, forcats

To install the complete tidyverse with:
`> install.packages("tidyverse")`

To load the core tidyverse with:
`> library(tidyverse)`

Useful Functions

> tidyverse_conflicts()	Conflicts between tidyverse and other packages
> tidyverse_deps()	List all tidyverse dependencies
> tidyverse_logo()	Get tidyverse logo, using ASCII or unicode characters
> tidyverse_packages()	List all tidyverse packages
> tidyverse_update()	Update tidyverse packages

dplyr

- dplyr is a grammar of data manipulation.
- It reduces repetition during analysis
- Reduces the probability of making errors
- Saves you some typing

Commonly used functions

1. select()
2. filter()
3. group_by()
4. summarize()
5. mutate()

Combine the verbs/functions with pipe operator %>%

Loading the data

```
> library(datasets)   #Load the datasets package

> library(gapminder)   #Load the gapminder package

> attach(iris)   #Attach iris data to the R search path
```

select()

- Allows you to select few variables among many in your dataset.

```
> sepal <- select(iris, Sepal.Length, Sepal.Width)
```

- Let's use the pipe operator

```
> sepal <- iris %>% select(Sepal.Length, Sepal.Width)
```

- To remove one column only from the dataset

```
> less_sepal <- select(sepal, -Sepal.Length)
```

- Renaming column names

```
> tidy_sepal <- iris %>%rename(sepal_width=Sepal.Width)
```

filter()

- allows you to select a subset of rows in a data frame.

```
> iris_virginica <- iris %>% filter(Species=="virginica")
```

```
>  iris_virginica_sepal <-  iris %>% filter(Species=="virginica", Sepal.Length > 6)
```

```
> iris_sesota_sepal <- iris %>% filter(Species=="setosa") %>%
select(Sepal.Length, Sepal.Width)
```

Challenge

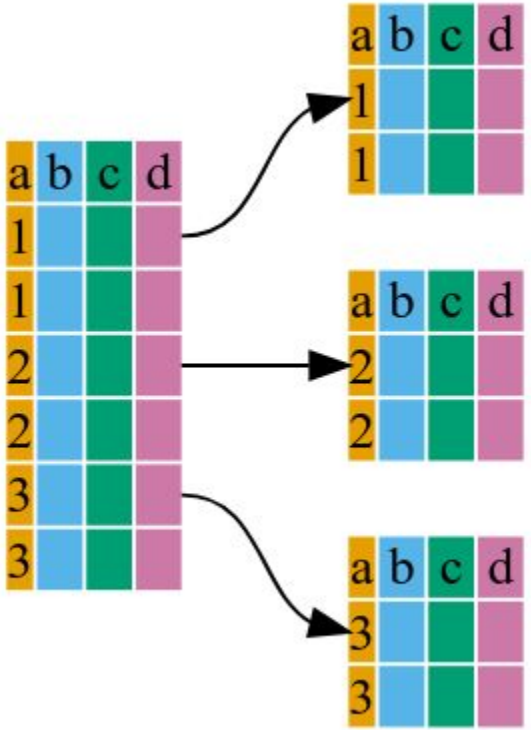
Use both `filter()` and `select()` to create a subset of data frame that contains sepal length of value more than 5, sepal width and species

group_by()

- Reduces error prone repetitiveness
- Split original data frame into multiple pieces

```
> str(iris)  #check the properties of the dataset
```

```
> str(iris %>% group_by(Species))
```



summarize ()

- Turn many observations into a single data point

To group per species and get the mean of Sepal.Width

```
> gdp_byspecies <- iris %>% group_by(Species) %>% summarize(mean_species = mean(Sepal.Width))
```

To group per two variable

```
> gdp_byspecies <- iris %>%  group_by(Species, Petal.Length) %>% summarize(mean_sepal_width = mean(Sepal.Width))
```

To calculate multiple single data points

```
> gdp_byspecies <- iris %>% group_by(Species, Petal.Length) %>% summarize(mean_sepal_width = mean(Sepal.Width), mean_petal_length mean(Petal.Length), sd_sepal_length = mean(Sepal.Length))
```

cont…

Challenge

Use `group_by()`, `summarize()`, `mean()`, `sd()`, `min()`, `max()` to calculate the mean, standard deviation, get maximum value, minimum value of each Species' Sepal.Width

mutate()

- updates or creates new variables of a data frame

Changing Sepal.Length to millimetre(update)

> `iris_SLMm_<-iris %>% mutate(Sepal.Length=Sepal.Length*10)`

Creating a new column called SLMm

> `iris %>% mutate(SLMm=Sepal.Length*10)`

Challenge

Use `group_by()`, `mutate()`, `summarize()`, `mean()`, `sd()`, `min()`, `max()` to calculate the mean, sd, find maximum and minimum of a new column of variable called SPlength where the Sepal.Length is divided by Petal.Length

ggplot2

- built on the grammar of graphics
- most effective for creating publication quality graphics
- Common plots
    a. Scatterplots
    b. Line Plots
    c. Bar plots
    d. Histogram
    e. Boxplot

Scatterplots

- Scatter plots allow you to compare two variables within your data.
- To do this with ggplot2, you use geom_point()

Create a subset of iris using Sepal.Length>5 as a condition

```
> iris_small <- iris %>% filter(Sepal.Length > 5)
```

Create a scatter plot that compares petal width and length

```
>  ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width)) + geom_point()
```

Adding color

```
> ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width, color=Species)) + geom_point()
```

Using different size for different variable

```
>  ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width, color=Species,
size=Sepal.Length)) + geom_point()
```

Scatterplot cont….

Faceting

```
> ggplot(iris_small, aes(x=Petal.Length, y=Petal.Width)) + geom_point()+
facet_wrap(~Species)
```

Line Plot

Create a sub-data

```
>  by_year <- gapminder %>% group_by(year) %>% summarize(medianGdpPerCap=median(gdpPercap))
```

Create a line plot

```
> ggplot(by_year, aes(x=year, y=medianGdpPerCap))+ geom_line()+ expand_limits(y=0)
```

Bar Plots

Create a sub data

```
> by_species <- iris %>% filter(Sepal.Length>6) %>% group_by(Species) %>% summarize(medianPL=median(Petal.Length))
```

Create a bar plot

```
>  ggplot(by_species, aes(x=Species, y=medianPL)) + geom_col()
```

Histogram

```
> ggplot(iris_small, aes(x=Petal.Length))+ geom_histogram()
```

Boxplot

```
> ggplot(iris_small, aes(x=Species, y=Sepal.Width))+ geom_boxplot()
```

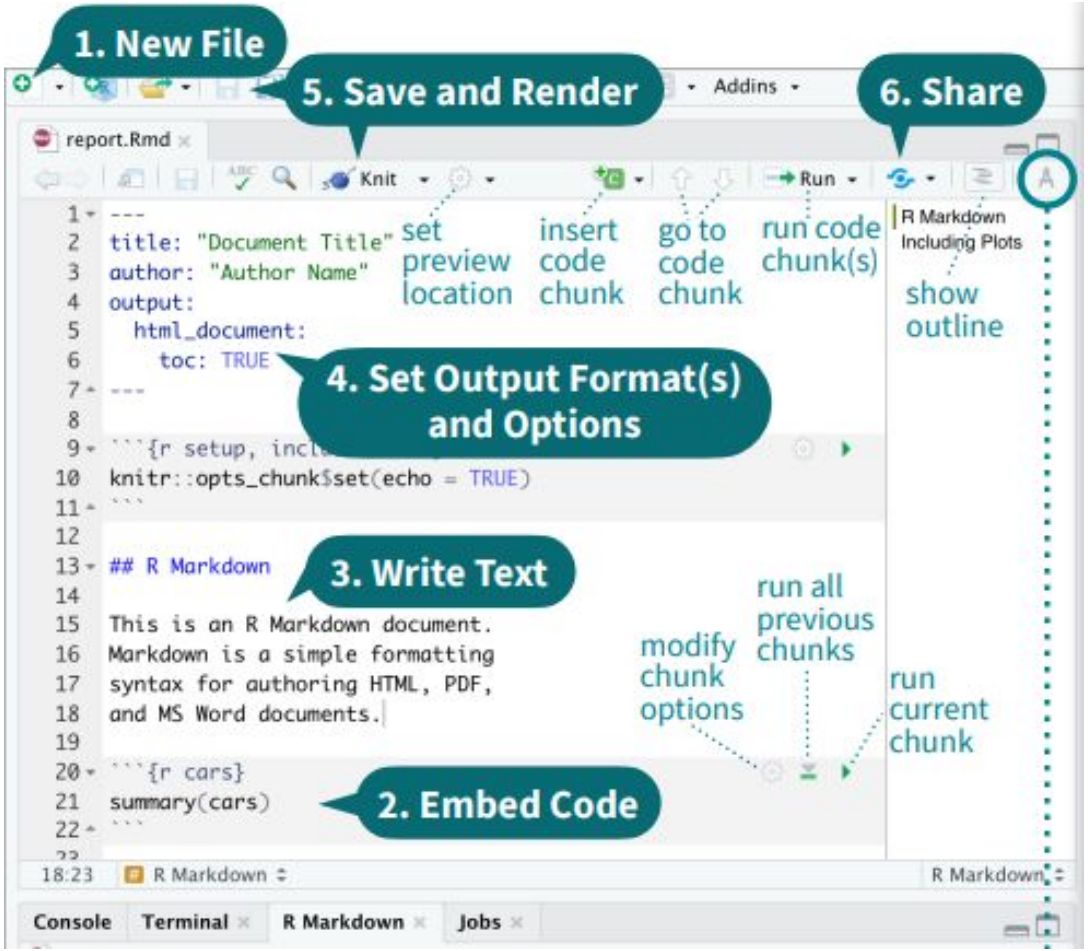
R markdown

- **Rmd files** · Develop your code and ideas side-by-side in a single document. Run code as individual chunks or as an entire document.

- **Dynamic Documents** · Knit together plots, tables, and results with narrative text. Render to a variety of formats like HTML, PDF, MS Word, or MS Powerpoint.

- **Reproducible Research** · Upload, link to, or attach your report to share. Anyone can read or run your code to reproduce your work

Interface


1. New File
5. Save and Render
6. Share
Addins
report.Rmd
Knit
Run
set preview location
insert code chunk
go to code chunk
run code chunk(s)
R Markdown Including Plots
show outline
4. Set Output Format(s) and Options
3. Write Text
run all previous chunks
modify chunk options
run current chunk
2. Embed Code
18:23
R Markdown
Console
Terminal
R Markdown
Jobs

Workflow

```
> install.packages("rmarkdown")
```

Open a new .Rmd file in the RStudio IDE by going to File > New File > R Markdown.

Embed code in chunks. Run code by line, by chunk, or all at once.

Write text and add tables, figures, images, and citations. Format with Markdown syntax or the RStudio Visual Markdown Editor.

Set output format(s) and options in the YAML header. Customize themes or add parameters to execute or add interactivity with Shiny.

Save and render the whole document. Knit periodically to preview your work as you write.

Share your work!

Shiny Apps

- A Shiny app is a web page (UI) connected to a computer running a live R session (Server)

```
> install.packages("shiny")
```

- App template

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)
```

Components of R Shiny App

User interface (ui) - nested R functions that assemble an HTML user interface for your app

Server - a function with instructions on how to build and rebuild the R objects displayed in the UI

ShinyApp - combines ui and server into a functioning app. Wrap with runApp() if calling from a sourced script or inside a function.

Demo

```
library(shiny)

ui <- fluidPage( numericInput(inputId = "n", "Sample size", value = 25),
plotOutput(outputId = "hist") )

server <- function(input, output) { output$hist <- renderPlot({ hist(rnorm(input$n)) })
}

shinyApp(ui = ui, server = server)
```

Reference

- Tidyverse manual
- Capentries https://swcarpentry.github.io/r-novice-gapminder/
- R markdown from R studio
- ShinyApp tutorials https://shiny.rstudio.com/tutorial/